

A Timed Calculus for Mobile Ad Hoc Networks

Mengying Wang

Software Engineering Institute
East China Normal University
Shanghai, China
mywang@sei.ecnu.edu.cn

Yang Lu

Department of Computer Science
Shanghai Jiaotong University
Shanghai, China
luyang0415@sjtu.edu.cn

We develop a timed calculus for Mobile Ad Hoc Networks embodying the peculiarities of local broadcast, node mobility and communication interference. We present a Reduction Semantics and a Labelled Transition Semantics and prove the equivalence between them. We then apply our calculus to model and study some MAC-layer protocols with special emphasis on node mobility and communication interference.

A main purpose of the semantics is to describe the various forms of interference while nodes change their locations in the network. Such interference only occurs when a node is simultaneously reached by more than one ongoing transmission over the same channel.

1 Introduction

Mobile ad hoc networks (MANETs) are complex distributed systems that consist of a collection of wireless mobile nodes that can dynamically self-organize into arbitrary network topologies, so as to allow people and devices to seamlessly interwork in areas without pre-existing communication infrastructures [1]. Owing to the flexibility and convenience, their applications have been extended from traditional military domain to a variety of commercial areas, e.g., ambient intelligence [2], personal area networks [4] and location-based services [3].

Wireless nodes use radio frequency channels to broadcast messages. Compared to the conventional wired-based broadcasts like Ethernet networks, this form of broadcast has some special features. First, broadcasting is *local*, i.e., a transmission covers only a limited area, called a *cell*, and hence reaches a (possibly empty) subset of the nodes in the network. Second, channels are *half-duplex*: on a given channel, a node can either transmit or receive, but cannot do both simultaneously. As a result, communication interference can only be detected at the destination. Further, nodes in MANETs can move arbitrarily, which makes the network easily suffer from interference. Since interference plays an important role in evaluating the performance of a network, it becomes a delicate aspect of MANETs that is handled by a great quantity of protocols (e.g., MACA/R-T[5]).

Over the last two decades, a number of process calculi have been proposed to model MANETs [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. These calculi can be divided into two categories according to their attentions to the network. The first group contains CBS# [6], CMAN [7, 10], RBPT [8], CNT [9], CMN [11], ω -calculus [13] and CSDT [15]. They attempt to depict *local broadcast* and *node mobility*. Take CMN as an example, each node is equipped with a location and a radius that define the cell over which the node can transmit. When a sender broadcasts messages, only nodes that are within its transmission cell could receive. Furthermore, nodes are marked mobile or stationary, and mobile nodes can change their locations randomly. Then CWS [12] and TCWS [14] constitute the second group. They focus on *local broadcast* and *communication interference*. The former abstracts the transmission into two state change events: begin transmission and end transmission, while the latter regards the transmission as a

time consuming procedure. To our knowledge, no calculus has integrated all of the three peculiarities, especially including node mobility and communication interference.

In this paper, we present a *timed calculus for mobile ad hoc networks* (TCMN), which extends CWS [12] and deals with all of the three issues. A central concern of our calculus is to describe the forms of interference while nodes move their locations in the network. Towards local broadcast, we write $n[Q]_{l,r}^c$ to stand for a node identified by n , located at l , executing process Q , and which can transmit data over channel c in a cell centered at l with radius r . As for node mobility, measures vary according to the specific situation. For instance, nodes that presently participate in no transmission could move arbitrarily without any impact on the environment. However, the movement of an active transmitter may affect the receptions of active receivers: some may get an error or interference, since they passively leave or enter the transmitter's transmission cell. Finally, with regard to communication interference, we assume all wireless nodes have been synchronized by some clock synchronization protocol [16, 17]. Then we consider a transmission proceeds in discrete steps which are represented by occurrences of a simple action σ to denote passing of one time unit. And if a receiver is exposed to more than one ongoing transmission over the same channel, it detects an interference.

In concurrency theory, Labelled Transition Semantics (LTS) is the most popular way of giving operational semantics since the transitions of a LTS expose the full behavior of the system (its internal activities as well as the interactions with the environment) which is required for defining behavioral equivalences and providing powerful proof techniques. However, sometimes the rules of a LTS may be difficult to understand particularly when the calculi relates to node mobility like [7, 10, 11]. Hence, a different form of operational semantics, named Reduction Semantics (RS), is introduced. RS only concerns the internal activities of a system, so it is easier to grasp. Besides, RS can be used to check the correctness of a LTS, by proving consistency with the LTS. For these reasons, we define both RS and LTS semantics for our TCMN and prove that they coincide.

We end this section with an outline of the paper. In Section 2, we define the syntax of our core language. Then in Section 3, we provide a RS for our calculus which specifies how an unbounded number of system components can be involved in an atomic interaction. Next a LTS that captures all the possible interactions of a term with its environment is proposed in Section 4. The equivalence between the RS and the LTS semantics is proved in Section 5. In Section 6 and 7, we extend our core language by adding some new operators to model some MAC-layer collision avoidance protocols: CSMA and MACA/R-T. We prove that the CSMA protocol doesn't solve the issue of node mobility while the MACA/R-T protocol is robust against node mobility. Finally, in section 8, we summarize our contributions and present the future work.

2 The Core Language

In Table 1, we present the core of TCMN. The syntax is defined in a two-level structure: a lower one for *processes* which describes the possible status of a node, and an upper one for *networks*. For easy understanding, in this section we only focus on those operators that are necessary for communication while the extended language will be presented in Section 6.

Generally, we use letters $a...c$ for channels, $m...o$ for identifiers, $x...z$ for variables, u for values that can be transmitted over channels: these include variables and closed values, and v for closed values, i.e. values that contain no variables. $\{u\}$ is a unary function designed to estimate the number of time units required for the transmission of the value u . Since only closed values will be used in transmissions, we assume the existence of an evaluation function $\llbracket \cdot \rrbracket$ to return the closed form of a value. Finally, we do not set how locations should be specified, the only assumption is that they should be comparable, so to

Table 1. The Syntax

<i>Networks:</i>			
$N \stackrel{\text{def}}{=} 0$	empty network	$ \ n[Q]_{l,r}^c$	node
$ \ N N$	parallel composition		
<i>Processes:</i>			
$Q \stackrel{\text{def}}{=} P$	non-active process	$ \ A$	active process
$P \stackrel{\text{def}}{=} 0$	termination	$ \ \text{out}\langle u \rangle.P$	output
$ \ \text{in}(x).P$	input		
$A \stackrel{\text{def}}{=} \langle v \rangle^\delta.P$	active output	$ \ (x)_v^\delta.P$	active input
<i>Values:</i>			
$u \stackrel{\text{def}}{=} x$	variable	$ \ v$	closed value
<i>Functions:</i>			
$f \stackrel{\text{def}}{=} \langle u \rangle$	time function	$ \ \llbracket u \rrbracket$	evaluation function
$ \ d(l_1, l_2)$	distance function		
where δ is a positive integer greater than 0			

determine whether a node is in or out of the transmission cell of another node. We do so by introducing a function d which takes two locations as parameters and returns the distance between them.

Networks are collections of nodes (which actually represent devices) that run in parallel and use same channels to communicate with each other. We use the symbol 0 to stand for the empty network, and $n[Q]_{l,r}^c$ to denote a node identified by n , located at l , executing process Q , and which can transmit data over channel c in a cell centered at l with radius r . We write $N|N$ to indicate a parallel composition of two sub-networks N .

Processes, living within the nodes, are sequential. For convenience, we divide processes into two categories: *non-active* and *active*. An active process is a process that is currently transmitting or receiving data, e.g., an active output process $\langle v \rangle^\delta.P$ denotes a transmitting process, and its transmission of value v will complete after δ time units. Similarly, an active input process $(x)_v^\delta.P$ represents a receiving process, and its reception of value v will last for the next δ instants of time. In the non-active process constructs, the symbol 0 stands for a terminated process. $\text{out}\langle u \rangle.P$ is an output process willing to broadcast the value $v = \llbracket u \rrbracket$, and once the transmission starts, the process evolves into the active output process $\langle v \rangle^\delta.P$, where $\delta = \langle u \rangle$ is the time necessary to transmit the value v . $\text{in}(x).P$ indicates an input process willing to receive data, and when the beginning of a transmission v in the following δ time units is captured clearly (i.e. without interference), the process becomes the active input process $(x)_v^\delta.P$. A node with an active output process inside is named *active transmitter*. Similarly, active input processes and non-active processes are included separately in *active receivers* and *non-active nodes*.

We assume that each node has a unique identifier, and different nodes cannot be located at the same position at the same time. We consider such networks *well-formed*. Since nodes cannot be created or destroyed, the well-formedness of a network is always preserved as the network evolves. In the remainder of the paper, all networks are well formed, and we use a number of notational conventions. Process Q stands for either a non-active or an active process while P and A represent non-active and active processes separately. We identify $\langle v \rangle^\delta.P = P$ and $(x)_v^\delta.P = P\{v/x\}$ if $\delta = 0$. We write $\text{out}\langle u \rangle$ for $\text{out}\langle u \rangle.0$, and $\langle v \rangle^\delta$ for $\langle v \rangle^\delta.0$.

3 Reduction Semantics

In this section, we study the reduction semantics (RS) for TCMN. In the literature [12], the only internal activity is a broadcast which is modelled by two events: begin transmission event and end transmission event. Yet in our system, a new type of internal activity: a migration is appended to depict node movement. In our model, the broadcast will be described by a begin transmission event and several time passing events (as shown in Table 2), while the migration will be represented by a node movement event from a specific node (as shown in Table 3). Among these three types of events, the *begin transmission event* (i.e. a node initiates a transmission) has the same meaning as that in [12], while the *time passing event* (i.e., a unit of time delays) is imported to replace the end transmission event in [12], and the *node movement event* (i.e., a node moves from one location to another) is a newly added event.

In our RS for core TCMN, a reduction denotes either a begin transmission event, or a time passing event, or a node movement event. In order to handle the interaction among an unbounded number of processes, we use rule schemas instead of simple rules to demonstrate the reductions. Also, since a reduction, e.g. a begin transmission event, cannot be performed inside arbitrary contexts: one should guarantee that the current context meets the specific conditions, the minimal information about the communication is attached to the reduction. Further, in order to model communication interference, we store all the needed active transmitters' information in a global set T which displays in any reduction to determine whether a node is simultaneously reached by more than one transmission over the same channel. For this reason, the reduction semantics is named RST: RS with parameter T . The component T is a set of triples (l, r, c) with each l, r, c in a triple represents location, radius and channel of an active transmitter separately. For simplicity, the semantics does not automatically update the set T . Therefore, when a reduction is performed, the new T which will be used in the next one has to be manually computed. However, it is not difficult to modify the rules so that they also produce the new T .

As usual in process calculi, the reduction semantics relies on an auxiliary relation, called *structural congruence*, denoted by \equiv , to allow the manipulation of the term structure so as to bring the participants of a potential interaction into contiguous positions. Here we define a smallest congruence including associativity, commutativity and identity over the empty network:

$$N|(N'|N'') \equiv (N|N')|N'' \quad N|N' \equiv N'|N \quad N|0 \equiv N$$

Next are some useful notations that will be used in RST:

- $T|_{l,c}$ is the subset of the active transmitters T whose transmissions are synchronized on channel c and can reach a node located at l . Formally,

$$T|_{l,c} = \{(l', r', c') | (l', r', c') \in T \wedge d(l', l) \leq r' \wedge c' = c\}$$

- $(l, r, c) \not\vdash_i N$ holds if Network N contains no input nodes $n[\text{in}(x).P]_{l',r'}^c$ or $n[(x)_v^\delta.P]_{l',r'}^c$ for which $d(l, l') \leq r$ is true (i.e., a transmission from a node located at l , with radius r , synchronized on c reaches no input nodes in N).
- $(l, r, c) \not\vdash_{ai} N$ holds if Network N contains no active input nodes $n[(x)_v^\delta.P]_{l',r'}^c$ for which $d(l, l') \leq r$ is true (i.e., a transmission from a node located at l , with radius r , synchronized on c reaches no active input nodes in N).

Let's explain the rules in Table 2 and 3. Rule RST-BEGIN is used to derive begin transmission reduction. As in [12], it rewrites atomically an output node $n[\text{out}(u).P]_{l,r}^c$ which is intending to initiate a transmission and all the receiver nodes that are not only in its transmission cell but also synchronized on

Table 2. Reduction Semantics - Begin transmission and time passing event

[RST-BEGIN]		[RST-PASS-NULL]
$\frac{\forall h \in I \cup J \cup K. d(l, h) \leq r \quad \forall i \in I. T _{l_i, c} = \emptyset \quad \forall j \in J. T _{l_j, c} \neq \emptyset}{T \triangleright n[\text{out}(u).P]_{l, r}^c \mid \prod_{h \in I \cup J} n_h[\text{in}(x_h).P_h]_{l_h, r_h}^c \mid \prod_{k \in K} n_k[(x_k)_{v_k}^{\delta_k}.P_k]_{l_k, r_k}^c \xrightarrow{\tau}_{l, r}^c}$		$T \triangleright 0 \xrightarrow{\sigma} 0$
$n[\langle \llbracket u \rrbracket \rangle^{\zeta u \delta}.P]_{l, r}^c \mid \prod_{i \in I} n_i[(x_i)_{v_i}^{\zeta u \delta}.P_i]_{l_i, r_i}^c \mid \prod_{j \in J} n_j[\text{in}(x_j).P_j]_{l_j, r_j}^c \mid \prod_{k \in K} n_k[P_k \{ \perp / x_k \}]_{l_k, r_k}^c$		
[RST-SENDING]		[RST-PASS-NA]
$\frac{\delta > 0 \quad \forall i \in I. d(l, l_i) \leq r}{T \triangleright n[\langle v \rangle^{\delta}.P]_{l, r}^c \mid \prod_{i \in I} n_i[(x_i)_{v_i}^{\delta}.P_i]_{l_i, r_i}^c \xrightarrow{\sigma} n[\langle v \rangle^{\delta-1}.P]_{l, r}^c \mid \prod_{i \in I} n_i[(x_i)_{v_i}^{\delta-1}.P_i]_{l_i, r_i}^c}$		$T \triangleright n[P]_{l, r}^c \xrightarrow{\sigma} n[P]_{l, r}^c$
[RST-CONT]	[RST-CONT-PASS]	[RST-CONGR]
$\frac{T \triangleright N \xrightarrow{\tau}_{l, r}^c N' \quad (l, r, c) \not\models_i N''}{T \triangleright N \mid N'' \xrightarrow{\tau}_{l, r}^c N' \mid N''}$	$\frac{T \triangleright N \xrightarrow{\sigma} N' \quad T \triangleright N'' \xrightarrow{\sigma} N'''}{T \triangleright N \mid N'' \xrightarrow{\sigma} N' \mid N'''}$	$\frac{N \equiv N' \quad T \triangleright N' \xrightarrow{\sigma} N'' \quad N'' \equiv N'''}{T \triangleright N \xrightarrow{\sigma} N''}$

the same channel c . After this reduction, the output process evolves into $\langle \llbracket u \rrbracket \rangle^{\zeta u \delta}.P$ indicating an active output process that will transmit the evaluation result $\llbracket u \rrbracket$ of value u in the following $\zeta u \delta$ time units. The effect of the begin transmission event on each receiver varies according to the structure of each receiver and the set T . There are three different situations, corresponding to the sets I , J and K . Processes in I represent normal inputs. Since their environments are silent ($T|_{l_i, c} = \emptyset$), they become active inputs of the form $(x_i)_{v_i}^{\zeta u \delta}.P_i$ and start receiving data $\llbracket u \rrbracket$ for the next $\zeta u \delta$ time units. By contrast, for processes in J , as they are currently reached by at least one other transmission ($T|_{l_j, c} \neq \emptyset$), they could not receive the begin transmission event clearly and stay idle. Finally, processes in K are active inputs, i.e., they are receiving another transmission, so the new begin transmission event causes interference, denoted by receiving symbol \perp .

Rule RST-SENDING deals with the time passing event for active processes. Initially, the active output process $\langle v \rangle^{\delta}.P$ requires δ time units to complete the data transmission. After a time interval, the remaining time would be $\delta - 1$ units for both sender and receivers. Meanwhile rule RST-PASS-NA and RST-PASS-NULL handle the time passing event for non-active processes and empty networks respectively. No matter how time flies, they remain unchanged.

Rule RST-MOVE-AO, RST-MOVE-AI1, RST-MOVE-AI2, RST-MOVE-AI3, and RST-MOVE-NA are all used to describe node movements. In RST-MOVE-AO, an active transmitter moves from l to l' . Then for active receivers in set I , as they are always reachable no matter from l or l' , they continue to receive data normally. As for active receivers in set J , since they are reachable from l but not from l' , they get an error, represented by a special sign ε . Finally, active receivers in set K , which are reachable from l' but not from l , are receiving another transmission, so the newly joined transmitter will make them get interference. Rule RST-MOVE-AI1, RST-MOVE-AI2 and RST-MOVE-AI3 depict all the different movements of an active receiver. In RST-MOVE-AI1, the active receiver moves from l to l' which makes the original transmission no longer receivable, hence it gets an error. While in RST-MOVE-AI2, although the active receiver moves from l to l' , it has always been in the transmitter's transmission cell and there is no more active transmitter in l' , so the active receiver remains unchanged. On the contrary, in RST-MOVE-AI3, when the active receiver arrives at l' , some other transmissions in l' interfere with its original one. As a result, the active receiver obtains an interference. Rule RST-MOVE-NA is straightforward, for

Table 3. Reduction Semantics - Node movement event

<p style="text-align: center;">[RST-MOVE-AO]</p> $\frac{\forall i \in I. d(l, l_i) \leq r \wedge d(l', l_i) \leq r \quad \forall j \in J. d(l, l_j) \leq r \wedge d(l', l_j) > r \quad \forall k \in K. d(l, l_k) > r \wedge d(l', l_k) \leq r}{T \triangleright n[(v)^\delta.P]_{l,r}^c \mid \prod_{h \in I \cup J} n_h[(x_h)^\delta_v.P_h]_{l_h,r_h}^c \mid \prod_{k \in K} n_k[(x_k)^\delta_k.P_k]_{l_k,r_k}^c \hookrightarrow_{l:l',r}^c n[(v)^\delta.P]_{l',r}^c \mid \prod_{i \in I} n_i[(x_i)^\delta_v.P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[P_j\{\varepsilon/x_j\}]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[P_k\{\perp/x_k\}]_{l_k,r_k}^c}$		
<p style="text-align: center;">[RST-MOVE-AI1]</p> $\frac{d(l, l_i) \leq r_i \wedge d(l', l_i) > r_i}{T \triangleright n[(x)^\delta_v.P]_{l,r}^c \mid n_i[(v)^\delta.P]_{l_i,r_i}^c \hookrightarrow n[P\{\varepsilon/x\}]_{l',r}^c \mid n_i[(v)^\delta.P]_{l_i,r_i}^c}$		
<p style="text-align: center;">[RST-MOVE-AI2]</p> $\frac{d(l, l_i) \leq r_i \wedge d(l', l_i) \leq r_i \quad T _{l,c} = T _{l',c}}{T \triangleright n[(x)^\delta_v.P]_{l,r}^c \mid n_i[(v)^\delta.P]_{l_i,r_i}^c \hookrightarrow n[(x)^\delta_v.P]_{l',r}^c \mid n_i[(v)^\delta.P]_{l_i,r_i}^c}$		
<p style="text-align: center;">[RST-MOVE-AI3]</p> $\frac{d(l, l_i) \leq r_i \wedge d(l', l_i) \leq r_i \quad \forall j \in J. d(l, l_j) > r_j \wedge d(l', l_j) \leq r_j \quad T _{l',c} = T _{l,c} \cup J}{T \triangleright n[(x)^\delta_v.P]_{l,r}^c \mid n_i[(v)^\delta.P]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[(v_j)^\delta_j.P_j]_{l_j,r_j}^c \hookrightarrow n[P\{\perp/x\}]_{l',r}^c \mid n_i[(v)^\delta.P]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[(v_j)^\delta_j.P_j]_{l_j,r_j}^c}$		
[RST-MOVE-NA]	[RST-CONT-MOVE]	[RST-CONT-INT]
$T \triangleright n[P]_{l,r}^c \hookrightarrow n[P]_{l',r}^c$	$\frac{T \triangleright N \hookrightarrow_{l:l',r}^c N' \quad (l, r, c) \not\models_{ai} N'' \wedge (l', r, c) \not\models_{ai} N''}{T \triangleright N N'' \hookrightarrow_{l:l',r}^c N' N''}$	$\frac{T \triangleright N \hookrightarrow N'}{T \triangleright N N'' \hookrightarrow N' N''}$

non-active nodes, their movement will not affect the environment, therefore they can move arbitrarily without any limitations and changes.

Rule RST-CONT, RST-CONT-PASS, RST-CONT-MOVE and RST-CONT-INT are closure rules with regard to different reduction forms ($\hookrightarrow_{l,r}^c$, \hookrightarrow^σ , $\hookrightarrow_{l:l',r}^c$, and \hookrightarrow). In RST-CONT, it provides a closure under contexts that do not contain receivers in the transmission cell of the transmitter. Similarly, rule RST-CONT-MOVE presents a closure under contexts that have never contained active receivers in the transmission cell of the transmitter when the transmitter moves from l to l' . Rule RST-CONT-INT is analogous to the previous two except that it concerns *internal events* (i.e., a node movement event from an active receiver or a non-active node). Rule RST-CONT-PASS is the time synchronization, it defines a closure under contexts that are also affected by the time passing event.

The last rule, RST-CONGR is a closure rule under structural congruence, where $\hookrightarrow^{\&}$ ranges over $\hookrightarrow_{l,r}^c$, \hookrightarrow^σ , $\hookrightarrow_{l:l',r}^c$ and \hookrightarrow for some c , l , l' and r .

4 Labelled Transition Semantics

We divide our Labelled Transition Semantics (LTS) into two set of rules corresponding to the two-level structure of our language. Table 4 contains the rules for the processes, while Table 5 and 6 presents those for the networks.

In the process semantics, a transition has the form $Q \xrightarrow{\alpha} Q'$, where the grammar for α is:

$$\alpha := !v : \delta \mid ?v : \delta \mid ?\perp \mid ?\varepsilon \mid \sigma$$

Table 4. Labelled Transitions for Processes

$\frac{\llbracket u \rrbracket = v \quad ?u = \delta}{\text{out}(u).P \xrightarrow{!v:\delta} \langle v \rangle^\delta.P} [\text{PS-OUT}_{begin}]$	$\frac{\delta > 0}{\langle v \rangle^\delta.P \xrightarrow{\sigma} \langle v \rangle^{\delta-1}.P} [\text{PS-OUT}_{send}]$
$\frac{-}{\text{in}(x).P \xrightarrow{?v:\delta} \langle x \rangle^\delta.P} [\text{PS-IN}_{begin}]$	$\frac{\delta > 0}{\langle x \rangle^\delta.P \xrightarrow{\sigma} \langle x \rangle^{\delta-1}.P} [\text{PS-IN}_{receive}]$
$\frac{-}{\text{in}(x).P \xrightarrow{?\perp} \text{in}(x).P} [\text{PS-IN}_{wait}]$	$\frac{-}{\langle x \rangle^\delta.P \xrightarrow{?\perp} P\{\perp/x\}} [\text{PS-IN}_{interfere}]$
$\frac{-}{\langle x \rangle^\delta.P \xrightarrow{?e} P\{e/x\}} [\text{PS-IN}_{err}]$	$\frac{-}{P \xrightarrow{\sigma} P} [\text{PS-PASS}]$
$\frac{\alpha \in \{?v:\delta, ?e, ?\perp\} \quad Q \notin \text{IQ}}{Q \xrightarrow{\alpha} Q} [\text{PS-NOIN}]$	

where IQ is the set of processes of the form $\text{in}(x).P$ or $\langle x \rangle^\delta.P$

Label $!v : \delta$ represents a begin transmission event (i.e., a transmission of value v in the following δ time units) is initiated by Q which then evolves into Q' ; $?v : \delta$ indicates a begin transmission event reaches Q and makes the process transform into Q' ; Analogously, $?\perp$ and $?e$ stand for an interference or error arrives; finally, σ means a time passing event.

Explanations for the rules in Table 4 are as follows: in PS-OUT_{begin} , the output process calculates the value u and initiates the transmission of the result v in the next δ time units; in PS-IN_{begin} , the input process successfully becomes involved with the transmission of value v for the next δ instants of time; in PS-OUT_{send} and $\text{PS-IN}_{receive}$, with the time passing by, the remaining transmission time is decreasing; in PS-IN_{wait} , the input process stays idle since it could not receive the begin transmission event clearly; in $\text{PS-IN}_{interfere}$ and PS-IN_{err} , an active input process encounters an interference or error in its reception, and hence stops receiving; Rule PS-PASS shows that the non-active process would never change as time goes by, and PS-NOIN demonstrates that the non-input processes would never respond to the reception of events.

Following are some useful mathematical symbols in LTS:

- $d(l, l') \leq r' \odot d(l, l'') \leq r' = (d(l, l') \leq r' \wedge d(l, l'') \leq r') \vee (d(l, l') > r' \wedge d(l, l'') > r')$.
- $T|_{l,c} - T|_{l',c}$ is the set of elements that are contained in $T|_{l,c}$ but not in $T|_{l',c}$.
- $T|_{l,c} \subset T|_{l',c}$ holds only if $T|_{l,c}$ is a proper subset of $T|_{l',c}$.

In the network semantics, transitions are of the form $T \triangleright N \xrightarrow{\mu} N'$ where T is the same as in Section 3. Let's comment on the rules in Table 5 and 6. Rule NS-OUT , NS-IN_1 , NS-IN_2 and NS-IN_3 concern the communication between a transmitter and its receivers. Rule NS-OUT shows that a node initiates a transmission. Then rule NS-IN_1 describes the behavior of a node that is within the transmission cell and could hear the begin transmission event clearly, whereas NS-IN_2 handles those that detect conflicts. Rule NS-IN_3 demonstrates that a node would not react to transmissions that are beyond its reception range or not in its listening channel.

Next rules NS-MOVE_{ao} , NS-MOVE_{in1} , NS-MOVE_{in2} , NS-MOVE_{in3} , NS-MOVE_{ai1} , NS-MOVE_{ai2} , NS-MOVE_{ai3} and NS-MOVE_{na} are all used to deal with the node movement events from different kinds of nodes. For example, rule NS-MOVE_{ao} depicts that an active transmitter located at l moves to l' during its transmission over channel c with radius r . Then the behaviors of surrounding nodes can be divided

Table 5. Labelled Transitions for Networks - Begin transmission and time passing event

$\frac{P \xrightarrow{!v:\delta} A}{T \triangleright n[P]_{l,r}^c \xrightarrow{c!v:\delta[l,r]} n[A]_{l,r}^c} [\text{NS-OUT}]$	$\frac{Q \xrightarrow{?v:\delta} Q' \quad d(l,l') \leq r' \quad T _{l,c} = \emptyset}{T \triangleright n[Q]_{l,r}^c \xrightarrow{c?v:\delta[l',r']} n[Q']_{l,r}^c} [\text{NS-IN}_1]$
$\frac{Q \xrightarrow{?v:\delta} Q' \quad d(l,l') \leq r' \quad T _{l,c} \neq \emptyset}{T \triangleright n[Q]_{l,r}^c \xrightarrow{c?v:\delta[l',r']} n[Q']_{l,r}^c} [\text{NS-IN}_2]$	$\frac{d(l,l') > r' \vee c \neq c'}{T \triangleright n[Q]_{l,r}^c \xrightarrow{c?v:\delta[l',r']} n[Q]_{l,r}^c} [\text{NS-IN}_3]$
$\frac{Q \xrightarrow{\sigma} Q'}{T \triangleright n[Q]_{l,r}^c \xrightarrow{\sigma} n[Q']_{l,r}^c} [\text{NS-PASS}]$	$\frac{-}{T \triangleright 0 \xrightarrow{c?v:\delta[l,r]} 0} [\text{NS-NULL}_{in1}]$
$\frac{-}{T \triangleright 0 \xrightarrow{\sigma} 0} [\text{NS-NULL}_{pass}]$	$\frac{T \triangleright N_1 \xrightarrow{c?v:\delta[l,r]} N'_1 \quad T \triangleright N_2 \xrightarrow{c!v:\delta[l,r]} N'_2}{T \triangleright N_1 N_2 \xrightarrow{c!v:\delta[l,r]} N'_1 N'_2} [\text{NS-COM}]$
$\frac{T \triangleright N_1 \xrightarrow{c?v:\delta[l,r]} N'_1 \quad T \triangleright N_2 \xrightarrow{c?v:\delta[l,r]} N'_2}{T \triangleright N_1 N_2 \xrightarrow{c?v:\delta[l,r]} N'_1 N'_2} [\text{NS-COM}_{in}]$	$\frac{T \triangleright N_1 \xrightarrow{\sigma} N'_1 \quad T \triangleright N_2 \xrightarrow{\sigma} N'_2}{T \triangleright N_1 N_2 \xrightarrow{\sigma} N'_1 N'_2} [\text{NS-SYN}]$

into three different cases corresponding to NS-MOVE_{in1}, NS-MOVE_{in2} and NS-MOVE_{in3} respectively. (1) For non-active nodes and active receivers that are receiving over other channels or that are always in or out of the transmission cell, they will remain unchanged. (2) For active receivers that are originally within the transmission cell, but later beyond it, they will receive an error. (3) For active receivers that are in the reverse situation, they will get interference. Analogously, rule NS-MOVE_{ai1}, NS-MOVE_{ai3} and NS-MOVE_{ai2} have described the possible scenarios of an active receiver that moves from l to l' : (1) if the active receiver moves out of the transmission cell, it will obtain an error; (2) if the active receiver has always been within the transmission cell and there is no more transmission in l' , it will continue to receive data normally; (3) if there are more transmissions in l' apart from its original one, the active receiver will get interference. Finally, we can see from NS-MOVE_{na} that for non-active nodes, they can move arbitrarily without conditions and limitations.

Moreover, rule NS-PASS represents the responses of nodes as time goes by. Rule NS-NULL_{in1}, NS-NULL_{in2} and NS-NULL_{pass} allow the empty network to receive data and evolve with time. At last, the propagation of events through networks is portrayed by rule NS-COM, NS-MOVE, NS-INT, NS-COM_{in}, NS-MOVE_{in}, and NS-SYN. The first three denote that an event generated in a network is propagated to the parallel network; while the later ones indicate that two parallel networks receive the same event.

5 Harmony Theorem

The Harmony Theorem aims at proving that the LTS-based semantics coincides with the RST-based semantics. With this objective, the theorem has three parts. First, it shows that the structural congruence respects the LTS, i.e., application of structural congruence will not change the possible transitions. Then it demonstrates that the RST behaves the same as the LTS, i.e., each reduction in the RST has a corresponding transition in the LTS which makes the resulting networks structurally congruent. In the end, it testifies the converse also holds.

Before proving the theorem, there are some auxiliary lemmas that portray the shape of processes able to perform a particular labelled transition, and the shape of the derivative processes (see the Appendix).

Table 6. Labelled Transitions for Networks - Node movement event

$\frac{}{T \triangleright n[\langle v \rangle \delta . P]_{l,r}^c \xrightarrow{c![(l:l'),r]} n[\langle v \rangle \delta . P]_{l',r}^c} [\text{NS-MOVE}_{ao}]$	
$\frac{(Q \notin \text{AIQ}) \vee (c \neq c') \vee (d(l,l') \leq r' \odot d(l,l'') \leq r')}{T \triangleright n[Q]_{l,r}^c \xrightarrow{c'?(l':l''),r'} n[Q]_{l',r}^c} [\text{NS-MOVE}_{in1}]$	
$\frac{Q \in \text{AIQ} \quad Q \xrightarrow{?e} Q' \quad d(l,l') \leq r' \wedge d(l,l'') > r'}{T \triangleright n[Q]_{l,r}^c \xrightarrow{c'?(l':l''),r'} n[Q']_{l',r}^c} [\text{NS-MOVE}_{in2}]$	
$\frac{Q \in \text{AIQ} \quad Q \xrightarrow{? \perp} Q' \quad d(l,l') > r' \wedge d(l,l'') \leq r'}{T \triangleright n[Q]_{l,r}^c \xrightarrow{c'?(l':l''),r'} n[Q']_{l',r}^c} [\text{NS-MOVE}_{in3}]$	
$\frac{Q \in \text{AIQ} \quad Q \xrightarrow{?e} Q' \quad T _{l,c} - T _{l',c} = T _{l,c}}{T \triangleright n[Q]_{l,r}^c \xrightarrow{} n[Q']_{l',r}^c} [\text{NS-MOVE}_{ai1}]$	
$\frac{Q \in \text{AIQ} \quad Q \xrightarrow{? \perp} Q' \quad T _{l,c} \subset T _{l',c}}{T \triangleright n[Q]_{l,r}^c \xrightarrow{} n[Q']_{l',r}^c} [\text{NS-MOVE}_{ai2}] \quad \frac{Q \in \text{AIQ} \quad T _{l,c} = T _{l',c}}{T \triangleright n[Q]_{l,r}^c \xrightarrow{} n[Q]_{l',r}^c} [\text{NS-MOVE}_{ai3}]$	
$\frac{}{T \triangleright n[P]_{l,r}^c \xrightarrow{} n[P]_{l',r}^c} [\text{NS-MOVE}_{na}] \quad \frac{}{T \triangleright 0 \xrightarrow{c'[(l:l'),r]} 0} [\text{NS-NULL}_{in2}]$	
$\frac{T \triangleright N_1 \xrightarrow{c'[(l:l'),r]} N'_1 \quad T \triangleright N_2 \xrightarrow{c![(l:l'),r]} N'_2}{T \triangleright N_1 N_2 \xrightarrow{c![(l:l'),r]} N'_1 N'_2} [\text{NS-MOVE}] \quad \frac{T \triangleright N_1 \rightarrow N'_1}{T \triangleright N_1 N_2 \rightarrow N'_1 N_2} [\text{NS-INT}]$	
$T \triangleright N_2 N_1 \xrightarrow{c![(l:l'),r]} N'_2 N'_1 \quad T \triangleright N_2 N_1 \rightarrow N_2 N'_1$	
$\frac{T \triangleright N_1 \xrightarrow{c'[(l:l'),r]} N'_1 \quad T \triangleright N_2 \xrightarrow{c'[(l:l'),r]} N'_2}{T \triangleright N_1 N_2 \xrightarrow{c'[(l:l'),r]} N'_1 N'_2} [\text{NS-MOVE}_{in}]$	

where AIQ is the set of processes of the form $(x)_{\nu}^{\delta}.P$

Theorem 1 (Harmony Theorem). Let N be a network, and T a set of active transmitters.

- (1) If $T \triangleright N \xrightarrow{\mu} N'$ and $N \equiv N_1$, then there exists N'_1 such that $T \triangleright N_1 \xrightarrow{\mu} N'_1 \equiv N'$.
- (2) (a) If $T \triangleright N \xrightarrow{c}_{l:l',r} N'$, then $T \triangleright N \xrightarrow{c![(l:l'),r]} N'_1 \equiv N'$.
 (b) If $T \triangleright N \xrightarrow{c}_{l,r} N'$, then there are ν and δ such that $T \triangleright N \xrightarrow{c!\nu:\delta[l,r]} N'_1 \equiv N'$.
 (c) If $T \triangleright N \xrightarrow{\sigma} N'$, then $T \triangleright N \xrightarrow{\sigma} N'_1 \equiv N'$.
 (d) If $T \triangleright N \hookrightarrow N'$, then $T \triangleright N \rightarrow N'_1 \equiv N'$.
- (3) For each item in (2), the reverse also holds.

Proof. Now we prove the three points in sequence.

- (1) The equivalence is defined in terms of commutativity, associativity, and identity over the empty network. First commutativity is guaranteed since rule NS-COM, NS-MOVE and NS-INT are symmetric, and rule NS-COM_{in}, NS-MOVE_{in} and NS-SYS are self-symmetric. Identity over the empty network conserves since, owing to rule NS-NULL_{in1}, NS-NULL_{in2} and NS-NULL_{pass}, the

Table 7. Extended Syntax for Processes

$P \stackrel{\text{def}}{=} \dots$	old processes	$ \lceil \text{in}(x).P \rceil^t Q$	input with timeout
$ \sigma.P$	delay	$ \triangleright c.P$	channel switch
$ [e]Q_1, Q_2$	choice	$ H(\overrightarrow{u})$	recursion
where t is a positive integer greater than 0			

empty network can perform any labels of the form $\frac{c?v:\delta[l,r]}{\rightarrow}$, $\frac{c?[l:l'],r]}{\rightarrow}$, and $\frac{\sigma}{\rightarrow}$, which serve as neutral element of parallel composition. Finally as the operations for parallel composition are associative and the network structure is always preserved, therefore associativity is also ensured.

(2) As proofs for the four statements are similar, we only take (a) as an example.

(a) The proof is by rule induction on the derivation of $T \triangleright N \xrightarrow{c}_{l:l',r} N'$.

First we consider rule RST-MOVE-AO, the proof for this case is by induction on the size of $l \cup j \cup k$. The base case is $l \cup j \cup k = \emptyset$, using rule NS-MOVE_{ao}. In the inductive case, we randomly choose an element h from $l \cup j \cup k$. Remember that by the inductive hypothesis, we already have a transition with label $\frac{c?[l:l'],r]}{\rightarrow}$. Below are different cases according to which set h belongs to. Suppose $h \in l$, then we can apply rule NS-MOVE_{in1} since $d(l, l_i) \leq r \wedge d(l', l_i) \leq r$ from the premise of rule RST-MOVE-AO. Thus the desired transition can be proved using rule NS-MOVE. Suppose now $h \in j$, we can use rule PS-IN_{err} to derive $(x_j)_{\nu}^{\delta}.P_j \xrightarrow{?e} P_j\{\varepsilon/x_j\}$, and then rule NS-MOVE_{in2} to derive a transition with label $\frac{c?[l:l'],r]}{\rightarrow}$. Hence the desired format can be arrived using rule NS-MOVE. Finally suppose $h \in k$, we can use rule PS-IN_{interfere} to derive $(x_k)_{\nu_k}^{\delta_k}.P_k \xrightarrow{? \perp} P_k\{\perp/x_k\}$. This transition can be lifted up to the network level using rule NS-MOVE_{in3} because $d(l, l_k) > r \wedge d(l', l_k) \leq r$ from the precondition of rule RST-MOVE-AO. Similarly, using rule NS-MOVE, we can get the desired transition.

The proof is analogous for rule RST-CONT-MOVE, since $(l, r, c) \not\models_{ai} N'' \wedge (l', r, c) \not\models_{ai} N''$ ensures that all the active input nodes satisfy the conditions of NS-MOVE_{in1}. As for non-active input nodes, rule NS-MOVE_{in1} can also be applied.

Rule RST-CONGR can be simulated by the first part of the theorem.

(3) The proof for (a) and (b) are based on Lemma 2 and Lemma 4 respectively. The proof for (d) is straightforward. Now we consider the proof for (c).

The proof is based on Lemma 5. If $l \cup j \cup k = \emptyset$, the desired reduction can be derived using rule RST-PASS-NULL and RST-CONGR. Otherwise, for each element i in l and for each element k in K , first apply rule RST-SENDING and RST-PASS-NA respectively, then employ rule RST-CONT-PASS and RST-CONGR to get the desired reduction. \square

6 The Extended Language

So far we have considered the subset of TCMN with only the operators that are necessary for communication. Now we present some extensions: a series of processes are added in Table 7, while the syntax for others remains the same.

First of all, the input construct is replaced by the *input with timeout* construct in $\lceil \text{in}(x).P \rceil^t Q$. This process is waiting for receiving a value, if the value arrives before the end of the t time units, the process evolves into an active receiver; otherwise, the process continues as Q . Process $\sigma.P$ stands for sleeping

Table 8. Extended Reduction Semantics

[RST-BEGIN]			
$\frac{\forall h \in I \cup J \cup K. d(l, l_h) \leq r \quad \forall i \in I. T _{l_i, c} = \emptyset \quad \forall j \in J. T _{l_j, c} \neq \emptyset}{T \triangleright n[\text{out}(u).P]_{l, r}^c \mid \prod_{h \in I \cup J} n_h[\text{in}(x_h).P_h]_{l_h, r_h}^c \mid \prod_{k \in K} n_k[(x_k)_{v_k}^{\delta_k}.P_k]_{l_k, r_k}^c \hookrightarrow_{l, r}^c n[\langle \llbracket u \rrbracket \rangle^{\delta_u}].P]_{l, r}^c \mid \prod_{i \in I} n_i[(x_i)_{u_i}^{\delta_{u_i}}].P_i]_{l_i, r_i}^c \mid \prod_{j \in J} n_j[\text{in}(x_j).P_j]_{l_j, r_j}^c \mid \prod_{k \in K} n_k[P_k\{\perp/x_k\}]_{l_k, r_k}^c}$			
[RST-INPUT-DELAY]	[RST-TIMEOUT]	[RST-PASS-DELAY]	
$\frac{t > 1}{T \triangleright n[\text{in}(x).P]_{l, r}^c \hookrightarrow_{l, r}^{\sigma} n[\text{in}(x).P]_{l, r}^{t-1} Q]_{l, r}^c}$	$\frac{t = 1}{T \triangleright n[\text{in}(x).P]_{l, r}^c \hookrightarrow_{l, r}^{\sigma} n[Q]_{l, r}^c}$	$T \triangleright n[\sigma.P]_{l, r}^c \hookrightarrow_{l, r}^{\sigma} n[P]_{l, r}^c$	
[RST-PASS-NA]	[RST-SWITCH]	[RST-IF-TRUE]	[RST-IF-FALSE]
$\frac{P \notin \text{DIQ}}{T \triangleright n[P]_{l, r}^c \hookrightarrow_{l, r}^{\sigma} n[P]_{l, r}^c}$	$T \triangleright n[\triangleright c'.P]_{l, r}^c \hookrightarrow_{l, r} n[P]_{l, r}^{c'}$	$\frac{e = \text{true}}{T \triangleright n[[e]Q_1, Q_2]_{l, r}^c \hookrightarrow_{l, r} n[Q_1]_{l, r}^c}$	$\frac{e = \text{false}}{T \triangleright n[[e]Q_1, Q_2]_{l, r}^c \hookrightarrow_{l, r} n[Q_2]_{l, r}^c}$
where DIQ is the set of processes of the form $\sigma.P$ or $\text{in}(x).P]_{l, r}^t Q$			

for one time unit while $\triangleright c.P$ represents a process that decides to switch its communication channel to c , and then continues as P . The construct $[e]Q_1, Q_2$ behaves as Q_1 if $e = \text{true}$ and as Q_2 otherwise. Here, e is a boolean value expression. Finally, $H(\vec{u})$ denotes a process defined via a (possibly recursive) definition $H(\vec{x}) \stackrel{\text{def}}{=} Q$, with $|\vec{x}| = |\vec{u}|$, where \vec{x} contains all free variables of Q .

We only provide the addition of the new operators to the RST semantics, since this is the simpler one and the one that we will use in Section 7. However the operators can be introduced in a similar way into the LTS semantics.

Before updating the RST semantics, a new structural congruence rule is appended:

$$n[H(\vec{u})]_{l, r}^c \equiv n[Q\{\vec{u}/\vec{x}\}]_{l, r}^c \text{ if } H(\vec{x}) \stackrel{\text{def}}{=} Q \wedge |\vec{x}| = |\vec{u}|$$

The additional reduction rules are shown in Table 8. Rule RST-BEGIN is as before, except that input is substituted by input with timeout. In RST-TIMEOUT, a timeout fires if no reception has started before the end of the current instant of time. For processes of the form $\text{in}(x).P]_{l, r}^t Q$ and $\sigma.P$, rule RST-INPUT-DELAY and RST-PASS-DELAY model the sleeping for one time unit respectively. Rule RST-PASS-NA is a modification of the former one: non-active processes other than those of the form $\text{in}(x).P]_{l, r}^t Q$ and $\sigma.P$ have no reaction to the time passing event. Then the remaining rules are self-explanatory.

7 Case Study

We start this section by taking some MAC-layer protocols: CSMA and MACA/R-T as examples to show the expressiveness of our calculus.

7.1 Carrier Sense Multiple Access

The *Carrier Sense Multiple Access* (CSMA) scheme is a widely used MAC-layer protocol. In this protocol, each device senses the channel (physical carrier) before its transmission. If the channel is free, the sender starts the transmission immediately; otherwise the device keeps monitoring the channel until it becomes idle and then starts the transmission.

We can easily model the carrier sense action of CSMA scheme by the process defined below:

$$\text{Send}(l, c, u) \stackrel{\text{def}}{=} [T|_{l,c} = \emptyset] \text{out}(u), \sigma. \text{Send}(l, c, u)$$

Let us represent some reduction traces for a network where nodes adopt the CSMA protocol. These traces indicate that the CSMA protocol does not address the issue of node mobility. When an active transmitter moves to the reception range of an occupied receiver, any transmission of the intruding node may cause interference with the ongoing transmission. Similarly, if an active receiver moves in the transmission cell of another transmitter, the transmission of the new transmitter will also interfere with the original one. Further, interference may as well occur when different packages are targeted at the same receiver simultaneously.

Example 1 (Interference). This example represents an active transmitter n_3 moves to n'_3 during its com-

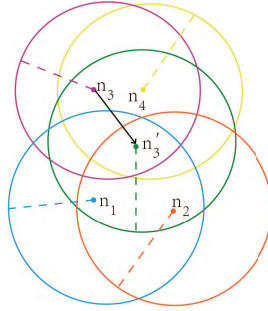


Figure 1: Network topology of Lemma 1-2 and Example 1-3

munication with node n_4 . Due to this event, the active receiver n_2 which is receiving data from node n_1 gets an interference since it passively enters the transmission cell of n'_3 . The network is:

$$N \stackrel{\text{def}}{=} n_1[(v_1)^{\delta_1}]_{l_1, r_1}^c \mid n_2[(x_2)^{\delta_1}.P_1]_{l_2, r_2}^c \mid n_3[(v_3)^{\delta_3}]_{l_3, r_3}^c \mid n_4[(x_4)^{\delta_3}.P_4]_{l_4, r_4}^c$$

where n_2 and n'_3 are in the transmission cell of n_1 , just as n_4 in n_3 , n_1 and n_2 together with n_4 in n'_3 (as shown in Fig.1).

We present a possible reduction trace, and it is easily understood.

$$\{(l_1, r_1, c), (l_3, r_3, c)\} \triangleright N \xrightarrow{c}_{l_3, l'_3, r_3} n_1[(v_1)^{\delta_1}]_{l_1, r_1}^c \mid n_2[P_2\{\perp/x_2\}]_{l_2, r_2}^c \mid n_3[(v_3)^{\delta_3}]_{l'_3, r_3}^c \mid n_4[(x_4)^{\delta_3}.P_4]$$

Analogously, if n_3 is an active receiver and moves to n'_3 during its reception from n_4 . Then the transmission of n_1 to n_2 will also interfere with n_4 to n'_3 . \square

Example 2 (Interference). This example indicates interference caused by the simultaneous transmission of two different packages. Let us consider now a different network:

$$N \stackrel{\text{def}}{=} n_1[\text{Send}(l_1, c, u_1)]_{l_1, r_1}^c \mid n_2[\text{Send}(l_2, c, u_2)]_{l_2, r_2}^c \mid n_3[\text{in}(x).P]_{l'_3, r_3}^c$$

where n_2 and n'_3 are in the transmission cell of n_1 , just as n_1 and n'_3 in n_2 (see Fig.1).

Table 9. MACA/R-T

$\begin{aligned} \text{SND}(\text{sid}, \text{rid}, u) &\stackrel{\text{def}}{=} \triangleright c_{r,\text{rid}}.\text{out}\langle\{\text{sid}, \text{rid}, \text{rts}, \{\{\text{sid}, \text{rid}, \text{end}, u\}\}\}\rangle. \\ &\quad \triangleright c_{s,\text{rid}}.\lceil \text{in}(x). \\ &\quad \quad [\text{fst}(x) = \text{rid} \wedge \text{snd}(x) = \text{sid} \wedge \text{trd}(x) = \text{cts}] \\ &\quad \quad \triangleright c_{s,\text{sid}}.\text{out}\langle\{\text{sid}, \text{rid}, \text{end}, u\}\rangle. \triangleright c_{r,\text{sid}}, \\ &\quad \quad \text{SND}(\text{sid}, \text{rid}, u) \rceil^t \\ &\quad \text{SND}(\text{sid}, \text{rid}, u) \end{aligned}$
$\begin{aligned} \text{RCV}(\text{id}, q) &\stackrel{\text{def}}{=} \lceil \text{in}(x). \\ &\quad [\text{snd}(x) = \text{id} \wedge \text{trd}(x) = \text{rts}] \\ &\quad \triangleright c_{s,\text{id}}.\text{out}\langle\{\text{id}, \text{fst}(x), \text{cts}, \text{fth}(x)\}\rangle. \\ &\quad \triangleright c_{s,\text{fst}(x)}.\lceil \text{in}(y). \\ &\quad \quad [\text{snd}(y) = \text{id} \wedge \text{fst}(y) = \text{fst}(x) \wedge \text{trd}(y) = \text{end}] \\ &\quad \quad \text{RCV}(\text{id}, \text{push}(q, \text{fth}(y))), \\ &\quad \quad \triangleright c_{r,\text{id}}.\text{RCV}(\text{id}, q) \rceil^t \\ &\quad \triangleright c_{r,\text{id}}.\text{RCV}(\text{id}, q), \\ &\quad \text{RCV}(\text{id}, q) \rceil^t \\ &\quad \text{RCV}(\text{id}, q) \end{aligned}$

A possible reduction trace is given:

$$\emptyset \triangleright N \hookrightarrow n_1[\text{out}\langle u_1 \rangle]_{l_1, r_1}^c | n_2[\text{Send}(l_2, c, u_2)]_{l_2, r_2}^c | n_3[\text{in}(x).P]_{l_3, r_3}^c \stackrel{\text{def}}{=} N_1$$

$$\emptyset \triangleright N_1 \hookrightarrow n_1[\text{out}\langle u_1 \rangle]_{l_1, r_1}^c | n_2[\text{out}\langle u_2 \rangle]_{l_2, r_2}^c | n_3[\text{in}(x).P]_{l_3, r_3}^c \stackrel{\text{def}}{=} N_2$$

Assign $v_1 = \llbracket u_1 \rrbracket$ and $\delta_1 = \langle u_1 \rangle$, then

$$\emptyset \triangleright N_2 \hookrightarrow_{l_1, r_1}^c n_1[\langle v_1 \rangle^{\delta_1}]_{l_1, r_1}^c | n_2[\text{out}\langle u_2 \rangle]_{l_2, r_2}^c | n_3[(x)^{\delta_1}.P]_{l_3, r_3}^c \stackrel{\text{def}}{=} N_3$$

Assign $v_2 = \llbracket u_2 \rrbracket$ and $\delta_2 = \langle u_2 \rangle$, then

$$\{(l_1, r_1, c)\} \triangleright N_3 \hookrightarrow_{l_2, r_2}^c n_1[\langle v_1 \rangle^{\delta_1}]_{l_1, r_1}^c | n_2[\langle v_2 \rangle^{\delta_2}]_{l_2, r_2}^c | n_3[P\{\perp/x\}]_{l_3, r_3}^c$$

Here n_1 senses the channel free, and then almost at the same time, n_2 also finds the channel available, so they begin to transmit data successively. Therefore an interference is generated at n_3' . \square

7.2 MACA/R-T

The *Receiver-Transmitter-Based Multiple Access with Collision Avoidance Protocol* (MACA/R-T) is a promising protocol used in MANETs. In MACA/R-T, all mobile nodes in the network agree to a set of pre-specified channels, e.g., a node id is assigned with $c_{r,\text{id}}$ and $c_{s,\text{id}}$ as its receiver and transmitter channels respectively. At the idle stage, all nodes will tune their receivers to their own receiver channel. When node sid wants to send a data package to node rid, node sid first sends a short control packet RST (request-to-send, which includes the sender id, the receiver id and the transmission duration of the data package) to node rid over channel $c_{r,\text{rid}}$ and then tunes its receiver to channel $c_{s,\text{rid}}$ to wait for a control packet CTS (clear-to-send, which includes the same duration information) from node rid. Upon receiving the RTS, node rid will send a CTS over channel $c_{s,\text{rid}}$ and tune its receiver to channel $c_{s,\text{sid}}$ for the data package. Finally, node sid receives the CTS and sends the data package to node rid over channel $c_{s,\text{sid}}$.

In Table 9, we provide an encoding of a sender and a receiver process in our TCMN with respect to the MACA/R-T protocol. We assume that the receiver has a queue to store the received packages, with

an operation push to insert an element. We also use four-tuples as values, with constructor $\{-, -, -, -\}$ and destructors fst, snd, trd and fth, retrieving the first, second, third and fourth component separately. We indicate with t the maximum time for a data package from the sender to arrive at the receiver.

The sender process $SND(sid, rid, u)$ runs at node sid and intends to transmit the value u to node rid . The process first switches its transmitter channel to $c_{r,rid}$ and then sends a RTS packet. After that, it waits for the CTS packet. If the CTS packet is not received before the end of the t time units, the process will move to itself and restart the transmission. On the other hand, if the CTS packet is received before the timeout, the data package $\{sid, rid, end, u\}$ is transmitted over channel $c_{s,sid}$ and the sender finishes the transmission.

The receiver process $RCV(id, q)$ is supposed to run at node id waiting for a RTS packet. If the RTS packet, with destination id , arrives before the timeout, the receiver switches its transmitter channel to $c_{s,id}$ and then replies with a CTS packet as well as waits for the data package over channel $c_{s,fst(x)}$. Otherwise, the receiver aborts the current reception and resets to process $RCV(id, q)$.

We show below that the MACA/R-T protocol is robust against node mobility, i.e., node movement will not give rise to communication interference. When an active transmitter moves to the reception range of an occupied receiver, the transmission of the intruding node will not interfere with the ongoing one. Besides, if an active receiver moves in the transmission cell of another transmitter, the transmission of the new transmitter will not interfere with the original one.

Lemma 1 Suppose when node n_1 is transmitting to node n_2 and node n_3 is transmitting to node n_4 , n_3 moves to n'_3 . The network topology is shown in Fig.1, n_2 and n'_3 are in the transmission cell of n_1 , just as n_4 in n_3 , n_1 and n_2 together with n_4 in n'_3 . Then the transmission of n'_3 to n_4 will not interfere with that of n_1 to n_2 .

Proof. Remember that only when an active receiver has received more than one transmission over the same channel, does the receiver get interference.

There are three kinds of packages in the MACA/R-T protocol: RTS, CTS and data, which are transmitted over channels $c_{r,rid}$, $c_{s,rid}$, and $c_{s,sid}$ respectively. According to the package types that n_1 and n_3 are sending, all the possible cases of the active transmitter n_3 moves to n'_3 are listed in the table below.

We can see that when an active transmitter (e.g., n_3) moves to the reception range of an occupied

$n_3 \dashrightarrow n_4$	$n'_3 \dashrightarrow n_4$	$n_1 \dashrightarrow n_2$	Interference at n_2	Reasons
$c_{r,n_4}.RTS$	$c_{r,n_4}.RTS$	$c_{r,n_2}.RTS$	No	Different channels
$c_{r,n_4}.RTS$	$c_{r,n_4}.RTS$	$c_{s,n_2}.CTS$	No	Different channels
$c_{r,n_4}.RTS$	$c_{r,n_4}.RTS$	$c_{s,n_1}.data$	No	Different channels
$c_{s,n_4}.CTS$	$c_{s,n_4}.CTS$	$c_{r,n_2}.RTS$	No	Different channels
$c_{s,n_4}.CTS$	$c_{s,n_4}.CTS$	$c_{s,n_2}.CTS$	No	Different channels
$c_{s,n_4}.CTS$	$c_{s,n_4}.CTS$	$c_{s,n_1}.data$	No	Different channels
$c_{s,n_3}.data$	$c_{s,n_3}.data$	$c_{r,n_2}.RTS$	No	Different channels
$c_{s,n_3}.data$	$c_{s,n_3}.data$	$c_{s,n_2}.CTS$	No	Different channels
$c_{s,n_3}.data$	$c_{s,n_3}.data$	$c_{s,n_1}.data$	No	Different channels

receiver (e.g., n_2), due to the different transmission channels, the transmission of the intruding node will not interfere with the ongoing one. \square

Lemma 2 Suppose when node n_1 is transmitting to node n_2 and node n_4 is transmitting to node n_3 , n_3 moves to n'_3 . As shown in Fig.1, n_2 and n'_3 are in the transmission cell of n_1 , just like n_3 and n'_3 in n_4 . Then the transmission of n_1 to n_2 will not interfere with that of n_4 to n'_3 .

Proof. The proof is similar to the one for Lemma 1, and we conclude that if an active receiver (e.g., n_3)

moves in the transmission cell of another transmitter (e.g., n_1), the transmission of the new transmitter will not interfere with the original one. \square

Nevertheless, in the MACA/R-T protocol, interference may still occur when different RTS packets are targeted at the same receiver simultaneously.

Example 3 (Interference). Let's consider the network:

$$N \stackrel{\text{def}}{=} n_1[\text{SND}(n_1, n_3, u_1)]_{l_1, r_1}^{c_{r, n_1}} \mid n_2[\text{SND}(n_2, n_3, u_2)]_{l_2, r_2}^{c_{r, n_2}} \mid n_3[\text{RCV}(n_3, [])]_{l'_3, r_3}^{c_{r, n_3}}$$

where n_2 and n'_3 are in the transmission cell of n_1 , as n_1 and n'_3 in n_2 (see Fig.1).

Here we present a possible reduction trace:

$$\emptyset \triangleright N \xrightarrow{\quad} \xrightarrow{\quad} \xrightarrow{c_{r, n_3}}_{l_1, r_1} \xrightarrow{c_{r, n_3}}_{l_2, r_2}$$

Initially, n_1 tunes its receiver to channel c_{r, n_3} and sends a RTS packet to n'_3 . Almost at the same time, n_2 also tunes its receiver to channel c_{r, n_3} and sends a RTS packet to n'_3 which unfortunately results in an interference at n'_3 . \square

8 Conclusions and Future Work

In this paper, we have proposed a timed calculus for mobile ad hoc networks paying particular attention to local broadcast, node mobility and communication interference. Then the operational semantics of our calculus is given both in terms of a Reduction Semantics and in terms of a Labelled Transition Systems. We have also proved that these two semantics coincide. Finally, we extend our core language by adding some new operators to model the CSMA and MACA/R-T protocol. And we have demonstrated that the former doesn't address the issue of node mobility while the latter is robust against node mobility.

In the future, a quantity of developments are possible. First, we would try to establish adequate Behavioral Equivalences which define when two terms have the same observable behavior. One possible approach is via UTP method, so as to investigate the denotational semantics for mobile ad hoc networks. Second, we would also like to study a set of algebraic laws, which can represent the features of mobile ad hoc networks.

References

- [1] Imrich Chlamtac, Marco Conti and Jennifer J.-N. Liu, *Mobile ad hoc networking: imperatives and challenges*, Ad Hoc Networks 1 (1) (2003) 13–64, doi:10.1016/S1570-8705(03)00013-1.
- [2] Jari Ahola, *Ambient Intelligence: Plenty of Challenges by 2010*, EDBT 2002, 2287 (2002) 14, doi:10.1007/3-540-45876-X_3.
- [3] Stefano Basagni, Imrich Chlamtac and Violet R. Syrotiuk, *Location aware one-to-many communication in mobile multi-hop wireless networks*, in: Proceedings of the IEEE Vehicular Technology (VTC), (2000), doi:10.1109/VETECS.2000.851464.
- [4] Thomas Guthrie Zimmerman, *Personal Area Networks (PAN): Near-field intrabody communication*, IBM Systems Journal 35 (3–4) (1996) 609–617, doi:10.1147/sj.353.0609.
- [5] Mario Joa-Ng and I-Tai Lu, *Spread Spectrum Medium Access Protocol with Collision Avoidance in Mobile Ad-hoc Wireless Network*, in: Proceedings of the IEEE INFOCOM 99, (1999) 21–25, doi:10.1109/INFCOM.1999.751465.

- [6] Sebastian Nanz and Chris Hankin, *A framework for security analysis of mobile wireless networks*, Theoretical Computer Science 367 (1–2) (2006) 203–227, doi:10.1016/j.tcs.2006.08.036.
- [7] Jens Chr. Godskesen, *A calculus for mobile ad hoc networks*, in: COORDINATION, in: Lecture Notes in Computer Science, 4467 (2007) 132–150, doi:10.1007/978-3-540-72794-1_8.
- [8] Fatemeh Ghassemi, Wan Fokkink and Ali Movaghar, *Restricted broadcast process theory*, in: SEFM, IEEE Computer Society, (2008) 345–354, doi:10.1109/SEFM.2008.25.
- [9] Fatemeh Ghassemi, Wan Fokkink and Ali Movaghar, *Equational reasoning on ad hoc networks*, in: FSEN, in: Lecture Notes in Computer Science, 5961 (2009) 113–128, doi:10.1007/978-3-642-11623-0_6.
- [10] Jens Chr. Godskesen, *A calculus for mobile ad-hoc networks with static location binding*, Electr. Notes Theor. Comput. Sci. 242 (1) (2009) 161–183, doi:10.1016/j.entcs.2009.06.018.
- [11] Massimo Merro, *An observational theory for mobile ad hoc networks (full version)*, Information and Computation 207 (2) (2009) 194–208, doi:10.1016/j.ic.2007.11.010.
- [12] Ivan Lanese and Davide Sangiorgi, *An operational semantics for a calculus for wireless systems*, Theoretical Computer Science 411 (19) (2010) 1928–1948, doi:10.1016/j.tcs.2010.01.023.
- [13] Anu Singh, C.R. Ramakrishnan and Scott A. Smolka, *A process calculus for mobile ad hoc networks*, Science of Computer Programming, 75 (6) (2010) 440–469, doi:10.1016/j.scico.2009.07.008.
- [14] Massimo Merro, Francesco Ballardin and Eleonora Sibilio, *A Timed Calculus for Wireless Systems*, Theoretical Computer Science 412 (47) (2011) 6585–6611, doi:10.1016/j.tcs.2011.07.016.
- [15] Dimitrios Kouzapas and Anna Philippou, *A process calculus for dynamic networks*, in: FMOODS/FORTE, in: Lecture Notes in Computer Science, 6722 (2011) 213–227, doi:10.1007/978-3-642-21461-5_14.
- [16] Qun Li and Daniela Rus, *Global clock synchronization in sensor networks*, IEEE Transactions on Computers 55 (2) (2006) 214–226, doi:10.1109/TC.2006.25.
- [17] Suyoung Yoon, Chanchai Veerarittiphan and Mihail L. Sichitiu, *Tiny-sync: tight time synchronization for wireless sensor networks*, ACM Transactions on Sensor Networks 3 (2) (2007) 81–118, doi:10.1145/1240226.1240228.

A Appendix

Lemma 1. If $T \triangleright N \xrightarrow{c?[(l:l'),r]} N'$, then

$$N \equiv \prod_{i \in I} n_i[(x_i)_{\vee}^{\delta}.P_i]_{l_i, r_i}^c \mid \prod_{j \in J} n_j[(x_j)_{\vee}^{\delta}.P_j]_{l_j, r_j}^c \mid \prod_{k \in K} n_k[(x_k)_{\vee}^{\delta_k}.P_k]_{l_k, r_k}^c \mid N''$$

where $\forall i \in I. d(l, l_i) \leq r \wedge d(l', l_i) \leq r$, $\forall j \in J. d(l, l_j) \leq r \wedge d(l', l_j) > r$, $\forall k \in K. d(l, l_k) > r \wedge d(l', l_k) \leq r$

and $(l, r, c) \not\models_{ai} N'' \wedge (l', r, c) \not\models_{ai} N''$. Furthermore

$$N' \equiv \prod_{i \in I} n_i[(x_i)_{\vee}^{\delta}.P_i]_{l_i, r_i}^c \mid \prod_{j \in J} n_j[P_j\{\varepsilon/x_j\}]_{l_j, r_j}^c \mid \prod_{k \in K} n_k[P_k\{\perp/x_k\}]_{l_k, r_k}^c \mid N''$$

Proof. The proof is by rule induction on the derivation of $T \triangleright N \xrightarrow{c?[(l:l'),r]} N'$.

Rule NS-MOVE_{in1} From the premise of the rule, we know that either $n[Q]_{l,r}^c$ is a non-active input node or an active input node that always within or beyond the mobile transmitter's transmission range. In the first and third case, the corresponding node can be inserted into N'' , while in the second case, $Q = Q' = (x_i)_{\vee}^{\delta}.P_i$, thus it follows that $I = \{i\}$, $J = K = \emptyset$, and $N'' = 0$.

Rule NS-MOVE_{in2} Here too we know that Q is an active input process and $Q \xrightarrow{? \varepsilon} Q'$, thus by inspection on the LTS for processes, we get one case, rule PS-IN_{err}. It corresponds to index Q with $j \in J$.

Rule NS-MOVE_{in3} Similarly, when Q is an active input process, then $Q \xrightarrow{? \perp} Q'$ can only be derived using rule RS-IN_{interfere}. Hence, Q is indexed with $k \in K$.

Rule NS-MOVE_{in} This is the inductive case. It brings the corresponding sets of indices and the non-index part of the network in the previous premises to the desired form. \square

Lemma 2. If $T \triangleright N \xrightarrow{c![(l:l'),r]} N'$, then

$$N \equiv n[\langle v \rangle^\delta . P]_{l,r}^c \mid \prod_{i \in I} n_i[(x_i)_v^\delta . P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[(x_j)_v^\delta . P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[(x_k)_{v_k}^{\delta_k} . P_k]_{l_k,r_k}^c \mid N''$$

where $\forall i \in I. d(l, l_i) \leq r \wedge d(l', l_i) \leq r$, $\forall j \in J. d(l, l_j) \leq r \wedge d(l', l_j) > r$, $\forall k \in K. d(l, l_k) > r \wedge d(l', l_k) \leq r$

and $(l, r, c) \not\ll_{ai} N'' \wedge (l', r, c) \not\ll_{ai} N''$. Furthermore

$$N' \equiv n[\langle v \rangle^\delta . P]_{l',r}^c \mid \prod_{i \in I} n_i[(x_i)_v^\delta . P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[P_j\{\varepsilon/x_j\}]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[P_k\{\perp/x_k\}]_{l_k,r_k}^c \mid N''$$

Proof. The proof is similar to the one for Lemma 1, and it uses Lemma 1 itself to handle premises which are input transitions. \square

Lemma 3. If $T \triangleright N \xrightarrow{c?v:\delta[l,r]} N'$, then

$$N \equiv \prod_{i \in I} n_i[\text{in}(x_i). P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[\text{in}(x_j). P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[(x_k)_{v_k}^{\delta_k} . P_k]_{l_k,r_k}^c \mid N''$$

where $\forall h \in I \cup J \cup K. d(l, l_h) \leq r$, $\forall i \in I. T|_{l_i,c} = \emptyset$, $\forall j \in J. T|_{l_j,c} \neq \emptyset$ and $(l, r, c) \not\ll_i N''$. Furthermore

$$N' \equiv \prod_{i \in I} n_i[(x_i)_v^\delta . P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[\text{in}(x_j). P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[P_k\{\perp/x_k\}]_{l_k,r_k}^c \mid N''$$

Proof. The proof is similar to the one for Lemma 1, using rules for begin transmission event. \square

Lemma 4. If $T \triangleright N \xrightarrow{c!v:\delta[l,r]} N'$, then

$$N \equiv n[\text{out}(u). P]_{l,r}^c \mid \prod_{i \in I} n_i[\text{in}(x_i). P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[\text{in}(x_j). P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[(x_k)_{v_k}^{\delta_k} . P_k]_{l_k,r_k}^c \mid N''$$

where $\forall h \in I \cup J \cup K. d(l, l_h) \leq r$, $\forall i \in I. T|_{l_i,c} = \emptyset$, $\forall j \in J. T|_{l_j,c} \neq \emptyset$ and $(l, r, c) \not\ll_i N''$. Furthermore if

$\llbracket u \rrbracket = v$ and $\langle u \rangle = \delta$ then

$$N' \equiv n[\langle v \rangle^\delta . P]_{l,r}^c \mid \prod_{i \in I} n_i[(x_i)_v^\delta . P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[\text{in}(x_j). P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[P_k\{\perp/x_k\}]_{l_k,r_k}^c \mid N''$$

Proof. The proof is similar to the one for Lemma 2, and it uses Lemma 3 itself to handle premises which are input transitions. \square

Lemma 5. If $T \triangleright N \xrightarrow{\sigma} N'$, then

$$N \equiv \prod_{i \in I} n_i[\langle v_i \rangle^{\delta_i} . P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[(x_j)_{v_j}^{\delta_j} . P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[P_k]_{l_k,r_k}^c$$

where $\forall i \in I. \delta_i > 0$ and $\forall j \in J. \delta_j > 0$. Furthermore

$$N' \equiv \prod_{i \in I} n_i[\langle v_i \rangle^{\delta_i-1} . P_i]_{l_i,r_i}^c \mid \prod_{j \in J} n_j[(x_j)_{v_j}^{\delta_j-1} . P_j]_{l_j,r_j}^c \mid \prod_{k \in K} n_k[P_k]_{l_k,r_k}^c$$

Proof. The proof is by induction on the size of $I \cup J \cup K$, and then for each case it is similar to the one for Lemma 1, using rules for time passing event. \square